3-8-2016

Learning Device Usage in Context: A Continuous and Hierarchical Smartphone Authentication Scheme

Bingyue Wang
Dartmouth College

Learning Device Usage in Context:

A Continuous and Hierarchical Smartphone Authentication Scheme

3/8/2016
Dartmouth Computer Science Technical Report TR2016-790
Bingyue Wang

Contents

Abstract

Popular smartphone authentication schemes, such as PIN-based or biometrics- based authentication methods, require only an initial login at the start of a usage session to authorize the user to use all the apps on the phone during the entire session. Those schemes fail to provide continuous protection of the smartphone after the initial login. They also fail to meet the hierarchy of security requirements for different apps under different contexts. In this study, we propose a continuous and hierarchical authentication scheme. We believe that a user's app-usage patterns depend on his location context. As such, our scheme relies on app-usage patterns in different location context to continuously establish the log probability density (LPD) of the authenticity of the current user. Based on different LPD thresholds corresponding to different security requirements, the current user either has a LPD higher than the threshold, which grants him continuous access to the phone or the app, or he has a LPD lower than the threshold, which locks him out of the phone or the app immediately. We test our scheme on 4,600 subjects from the Device Analyzer Dataset. We found that our scheme could correctly identify the authenticity of the majority of the subjects. However, app-usage patterns with or without location context yielded similar performances, indicating that user contexts did not contribute further information to establish user behavioral patterns. Based on our scheme, we propose a hypothetical Android app which would provide continuous and hierarchical authentication for the smartphone users.

1 Introduction

Recent market research has predicted that 2 billon consumers will have smartphones by 2016 [1]. Of these 2 billon smartphone users, many are using their phones to store an increasingly wider range of personal and sensitive information such as banking accounts, health data, intimate photos, personal conversations, romantic interests, and corporate secrets. Furthermore, many are also using their phones as identity tokens to access online services, such as social networks, as well as physical services, such as unlocking doors, controlling cars, boarding planes, and purchasing goods and services. Given the increasing amount of personal data that is stored on the phone and the growing number of sensitive actions that are available through the phone, there is an important need to design a mobile authentication method that is both usable and secure.

To date, the most popular mobile authentication methods are non-hierarchical and non-continuous. Non-hierarchical authentication means that the same authentication method authorizes the users to use all applications on the mobile phones regardless of differing sensitivity of different apps under different context [2]. Non-continuous authentication means that some *initial authentication* scheme, such as password or biometrics, is only prompted at the beginning of a usage session. Once the mobile phone is unlocked by the user through the initial authentication, the phone remains accessible to all people until either the user turns off the screen or the current usage session has timed out.

Non-hierarchical authentication provides poor user experience as it does not adapt to users' different security requirements for different apps and under different contexts. In one study, subjects indicated that they want half of their apps to remain accessible all the time, and the other

half to be protected by authentications [2]. Some subjects also indicated that they do not worry about device security when the device is left at home [2]. In light of these results, one research study proposed to cluster GPS data to determine whether the subjects are at home, at work or at other places, and trigger different authentication methods based on the *location context* [8].

Non-continuous authentication is also problematic as an adversary can easily steal valuable information on the phone after a user enters the initial authentication and leaves his phone unattended [3]. Previous research suggested continuous authentication of mobile phones, perhaps by recognizing learned user behaviors. In these methods, sensor and usage data (measured by smartphones) are aggregated to establish a user *behavioral profile* to non-intrusively and continuously authenticate the user as he is using the phone. Some examples of user behaviors include touch screen interaction [4], phone calls [5], text messages [5], browsing history [5], gait [6], and typing patterns [8].

In our research, we aim to develop an authentication method that is both continuous and hierarchical. To improve on the existing research on continuous authentication, we augment the *behavioral profile* with *location context*. Our hypothesis is that the user behaviors are highly dependent on the location context. Therefore a hybrid model of user behaviors and location context would more securely and easily establish the identity of the correct owner. We define user behaviors to include app-usage data such as time of the day, number of apps used, duration of app-usage, and the top three apps used. We define location context to include location information indicated by either GPS coordinates or Wi-Fi MAC addresses.

In our approach, we first cluster each user's past location traces automatically to establish different contexts. We then build a behavioral model for each context for each user using past

behavioral data. This context-aware behavioral model is then used to calculate the log probability density (LPD) for the current smartphone usage; that is, an estimate of the likelihood that the current behavior and context fits the user's historical behavior for this context.

Our scheme can be used for both system-wide and app-specific access control. In terms of system-wide access control, our scheme provides continuous defense of the smartphone after the initial authentication. Once a user unlocks his phone through the initial authentication, a LPD value is continuously calculated. If at one moment the LPD value does not exceed a threshold specified in the user security settings, the phone will prompt the initial authentication again to authenticate the user for further usage of the phone. This continuous system-wide access control can be adapted to satisfy the different security needs under different contexts. For example, a user may specify a lower threshold for phone usage at home and a higher threshold for phone usage under unfamiliar contexts. In terms of app-specific access control, our approach allows users to specify different security needs for different apps. For example, for non-sensitive apps such as Calculator, Weather, and News, a user can specify a low threshold for the LPD such that no password will be required even if the user behavior does not fit the behavioral model in the context. For apps that contain or provide access to personal information such as Facebook, Mail, and Phone, a user can specify a medium level threshold for the LPD such that password will be required if the user behavior deviates too much from the context-aware behavioral model. For apps that contain highly sensitive information such as Banking, Shopping, and Health, a high threshold for the LPD might be specified to require that user behavior follows the model closely in the given context.

2 Related Work

Our research both drew inspiration from and improved on existing research about continuous authentication and hierarchical authentication.

2.1 Continuous Authentication

Existing research on continuous authentication relies on the collection of user behavioral data from the phone to "continuously and transparently authenticate mobile device users" [10]. Some of these behavioral data include touch-screen interaction [4], phone calls [5], text messages [5], browsing history [5], gait [6] and typing patterns [8].

However, simply relying on user behavioral data does not sufficiently and securely authenticate the smartphone user. First, many user behaviors do not provide a constant stream of data, and no authentication can be properly established during the period of no data. For example, sedentary users would have long periods where no gait information can be detected [10]. Typing activity will be scarce for users who mostly use their phones to play games, browse news, and make phone calls. Second, some of the behavioral pattern established by previous research may be susceptible to adversarial mimicry. For example, an adversary with possession of the smartphone can simply check browsing history to enter a few recent websites to replicate the authentic behavioral patterns to gain authorization [10]. Third, user behaviors vary in different contexts, but current research often generalizes a single behavioral pattern across all contexts [10] [11]. Such blanket behavioral patterns may yield good results during controlled experiments that require the users to perform certain actions, but the blanket pattern may not be "representative of natural user interactions" [11].

2.2 Hierarchical Authentication

One research direction for the hierarchical authentication is to use location context to determine the level of confidence in the current context and to trigger different initial authentication methods accordingly. For example, Hayashi et al. used location traces and 5-Nearest Neighbor classification algorithm to determine whether the subject is at home, at work or at other places [8]. Such context information is then used to trigger different initial authentication methods such as "no authentication, a PIN, [or] a password" [8]. During the five-day training phase of this scheme, users are required to label their location traces as either "home", "work", or "other". Such a long duration for explicit labeling of location traces raises not only usability issues but also privacy concerns.

Another research direction of hierarchical authentication is to use a combination of signals from the smartphone to determine the level of confidence in the user's authenticity. Riva et al. suggested the use of biometric signals, behavioral signals, possession signals and secrets to determine the different confidence levels of the user's authenticity [9]. Both the confidence levels of the user's authenticity and the security requirements of each app are labeled as one of "public", "private", or "confidential" [9]. A user can only access those apps with security requirements lower than or equal to the user's calculated confidence level. For example, a user who has a "private" confidence level can only access apps labeled as "public" or "private", but not apps labeled as "confidential" [9]. However, in the report of their results, 37.31% of confidential users were recognized as public, while 7.44% of public users were recognized as private.

2.3 Our Approach

Our approach provides the following improvements over the existing research. First, we analyze the user behavior in context. We do not rely on only the behavioral data, which vary across different contexts. Nor do we consider solely the contextual data, which are only a good second factor and not a primary factor in authentication [10]. The combination of behavioral and contextual data also enables us to build a more nuanced and complex user profile that is less susceptible to adversarial mimicry. Second, our model uses data that occur with high frequency and are almost always available. The high availability of our feature set enables us to achieve truly "continuous" authentication as new data are continuously fed into our context-aware behavioral model to authenticate the users. Third, we test our proposal on a large dataset of Android device usage data. The dataset was collected in uncontrolled environments. Thus it should avoid the pitfalls of "unauthentic usage" as in many controlled and lab-based experiments. Fourth, in our hierarchical approach, the users, rather than the researchers, can set different security requirements using different LPD thresholds to balance their needs for convenience and security.

3 Algorithm and Implementation

The development of our context-aware behavioral model had three essential phases, outlined in Figure 1 below and explained in more detail in the following sub-sections.

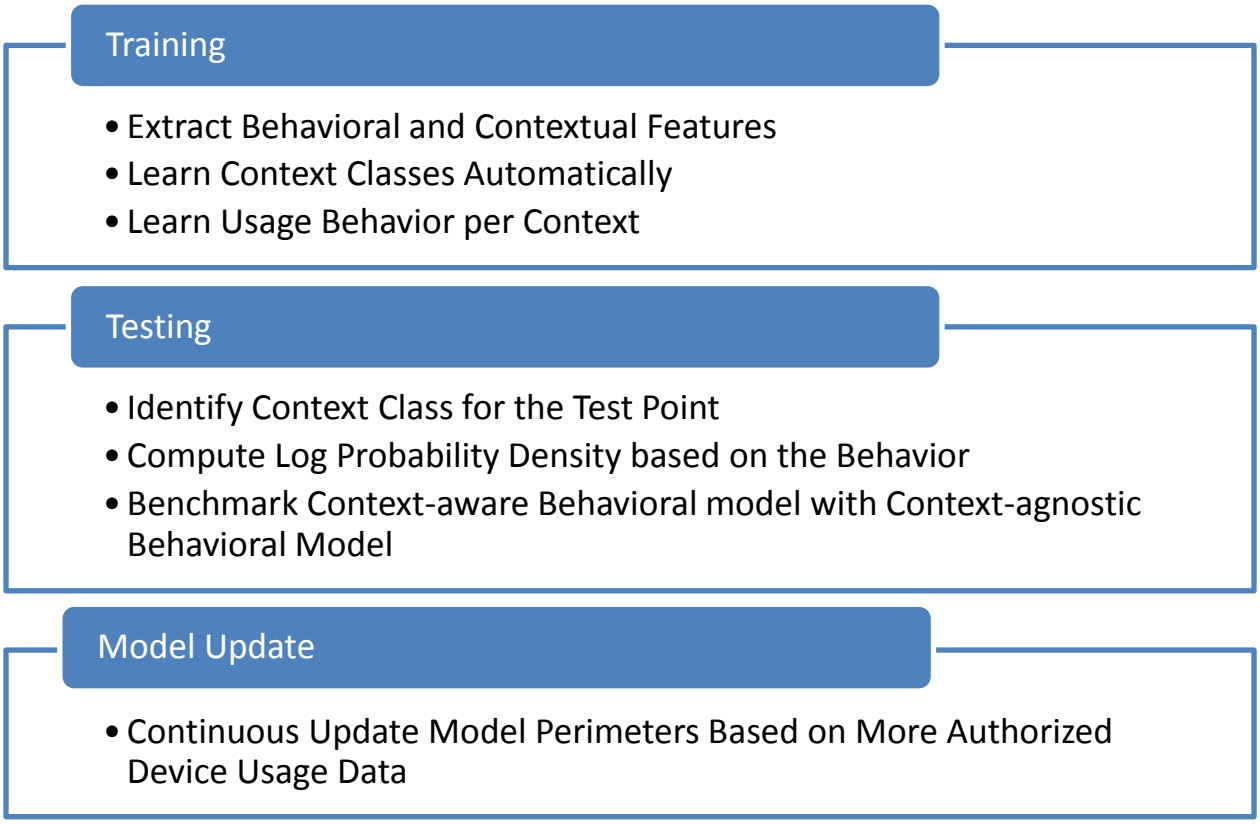

Figure 1 Flow Chart of Study Process

3.1 Device Analyzer Dataset

Both the training and the testing phase of our project used the Device Analyzer Dataset collected by Cambridge University. The Device Analyzer Dataset contains "over 100 billion records of Android smartphone usage from over 17,000 devices across the globe" [17]. We have chosen the Device Analyzer Dataset for three reasons. First, the Device Analyzer collects data for a large number of features, including all of the behavioral and the contextual features we need. Second, the Dataset contains information for a large number of subjects from many different geographical areas. The diversity and the scale of the dataset allow us to better predict how our model will perform in the real world. Third, subjects of the Device Analyzer study simply install the data collection app on their phones and continue to use their phones as normal. We expect that this naturalist data collection method provides more authentic user behaviors than a lab-based data collection method.

As user behaviors and contexts are likely to shift in the long run, we focus on a maximum of 21 days of data for each subject, as suggested by previous work [16]. The Device Analyzer Dataset collected different types of data at different frequency. As such, we combine one incidence of Wi-Fi scan with its closest subsequent incidence of app-usage data collection into one single data entry. We define one data entry for a subject to consist of two neighboring incidences of data collection, which must contain all of the behavioral and contextual features we require. Since not all subjects in the Device Analyzer Dataset have continuous stream of data we require, we select only those 4,600 subjects who have at least 1,000 data entries for their first 21 days of data. Appendix I gives further information about the data quality of the 4,600 chosen subjects.

For each of the 4,600 subjects, we divide sequentially all of his or her data entries in the proportion of 9 to 1 for the training and the testing of our models respectively.

3.2 Training Phase

During the training phase, we extract all of the required behavioral and contextual features from the training set and build a behavioral model for each subject in each context.

3.2.1 Extract Behavioral and Contextual Features

To extract behavioral features, we first loop through all data entries of each subject, and we rank the top three apps for each subject based on the total duration of usage of each app. We then extract the following features for each data entry for every subject, and we normalize all the following features so that they have comparable means and variances:

- Minute in the day at the present app data collection
- Total number of apps used at the foreground since the last app data collection
- Total duration of foreground app-usage since the last app data collection
- Duration of usage for the top 1 - 3 apps since the last app data collection

We selected these features because their data occur at a high frequency. Moreover, we believe those features vary in different contexts. For example, a user is more likely to use certain apps in the afternoon with a workplace Wi-Fi while he is more likely to use other apps at night with a home Wi-Fi. Furthermore, we also believe the app-usage features vary from individual to individual. For example, a heavy phone user might have a longer total duration of app-usage and/or likely a larger number of apps used. On the other hand, an infrequent phone user might have a shorter total duration of usage and/or fewer number of apps used.

To extract location context, we use the set of Wi-Fi MAC addresses available in each Wi-Fi scan. Most of previous research has used GPS coordinates as an indicator of location context. The use of GPS is not preferable in our opinion because storing GPS traces in the phones raises potential security problems if an adversary takes possession of the phone and discovers where the user is living through those GPS traces. Furthermore, GPS works only in open areas, but urban dwellers spend most of their time indoors. The basement and the top floor in the same building might be different contexts and thus we should expect different usage patterns for the same smartphone user. Most importantly, the Device Analyzer Dataset has only around 150 subjects who were willing to share their GPS traces. Using GPS coordinates for location context would considerably decrease the number of qualified subjects. Given the shortcomings of GPS coordinates, we decided to use Wi-Fi MAC addresses as a proxy for location. The Wi-Fi MAC addresses have been one-way hashed to better protect the users' privacy. Unlike GPS coordinates, Wi-Fi MAC addresses can often distinguish between the top floor and the basement within the same building.

3.2.2 Learning Context Classes

The second step in the training phase is to automatically cluster all contextual features into different context classes. We expect that different smartphone users will likely have different numbers of context classes. For example, a student may have three context classes: study space, dorm space, and other space. On the other hand, a frequent business traveller may have multiple context classes in each city where he works. Since the number of context classes is unknown and varies from individual to individual, we used DBSCAN to automatically find the number of context classes and to classify all contextual features into each of these classes [12].

The implementation of DBSCAN requires a distance function to gauge the differences between two sets of features. Since our location context is non-numeric and contains sets of Wi-Fi MAC addresses found during Wi-Fi scan, we believe that the optimal choice for distance function would be the Jaccard distance function, which is used for comparing similarity or diversity between different sets [13]. The pseudo code for the implementation of Jaccard distance function is given in Figure 2 below:

```
def get_jaccard_distance(set1, set2):
    return 1 - (len(set1.intersection(set2)) / (len(set1.union(set2))))
```

Figure 2 Jaccard Distance Pseudo Code

The implementation of DBSCAN also requires us to specify the *epsilon* value, which is the "maximum distance between two samples for them to be considered as in the same neighborhood", and *min_samples* value, which is the "number of samples in a neighborhood for a point to be considered as a core point" [14]. If a particular point does not have at least *min_samples* number of points in a distance *epsilon* from itself, this point will be considered as a "noisy point".

To select appropriate values for *epsilon* and *min_samples*, we selected 20 representative subjects who had at least 1,000 entries of Wi-Fi MAC addresses and GPS coordinates for the first 21 days of data collection. For each subject, we clustered his Wi-Fi MAC addresses using *epsilon* values increasing from 0.1 to 0.9 at an interval of 0.2, and *min_samples* values increasing from 60 to 150 at an interval of 30. We define the best *epsilon* and *min_samples* value pair to be the one which generated relatively fewer "noisy points" and produced a reasonable number of clusters (e.g. 1~5). After analyzing the results of all 400 (=20x5x4) runs of DBSCAN, we found the best value pair for the 20 representative subjects to be (*epsilon* = 0.7, *min_samples* = 150).

We assumed this value pair to be the best for all of the 4,600 subjects chosen from the Device Analyzer Dataset. Detailed scatter plots of the number of clusters and the number of "noisy points" against different *eps* and *min_samples* values can be seen in Appendix II.

To make sure our best value pair (*epsilon* = 0.7, *min_samples* = 150) generated Wi-Fi MAC address clusters that properly model the location context, we calculated the in-cluster and inter-cluster GPS Euclidian distances for all clusters generated by DBSCAN on Wi-Fi MAC addresses. The best value pair indeed gave smaller mean in-cluster GPS Euclidian distance and larger mean inter-cluster GPS Euclidian distance. This showed that DBSCAN with the best value pair clustered those Wi-Fi MAC addresses together if they were geographically close and separated those Wi-Fi MAC addresses into different clusters if they were geographically further apart.

3.2.3 Learning Usage Behavior per Context

We assume that each subject's behavioral features to consist of multiple normal distributions. Once DBSCAN successfully clustered all data entries into different context classes, we built for each context class a Gaussian Mixture Model (GMM) using the behavioral features extracted from all entries in that context class. A GMM is a parametric probability density function that calculates the probability density of the behavioral data entry x under the context C using N normal distributions, each with mean μ_i, covariance matrix Σ_i , and mixing weights w_i. The formula is shown below.

$$p(x|C) = \sum_{i=1}^{N} w_i \, N(x|\mu_i, \Sigma_i)$$

In the above formula, N can be any positive integer. The type of covariance parameter for Σ_i can be any one of "diagonal", "spherical", "tied", or "full". To find the best N and the type of

covariance parameter, we loop through different values (1 to 10) for N and four different types of covariance parameter. For each value of N and type of covariance parameter, we use a Python machine learning package, scikit-learn, to conduct expectation-maximization to find the mean μ_i, covariance matrix Σ_i , and mixing weights w_i of each of the N mixture components [15]. We then use Bayesian Information Criterion (BIC) to see how closely the GMM model represents the actual data. We eventually pick the value of N and type of covariance parameter that gives the best BIC results.

The entire algorithm for finding the optimal GMM for each context class is described in the pseudo code below. The initialization, expectation, and maximization steps described below are a summarized version of the expectation-maximization algorithm implemented by scikit-learn [15].

```
def generate_behavioral_model_per_context(behavioral_features_of_current_context):
    lowest_bic = infinity
    for covariance_type in ['spherical','tied','full','diagonal']:
        for n in range(1,10):
            n_iterations = 0
            #initializations
            cluster_centroids = k_means_find_centroids(behavioral_features_of_current_context,
                                       n_component = n)
            means = cluster_centroids
            variances = covariance(behavioral_features_of_current_context.T)
            weights = list_of_n_value_(1/n)
            previous_likelihood = negative_infinity

            while n_iterations < 100:
                n_iterations++

                # Expectation Step
                current_likelihood, responsibilities = log_likelihood_of_GMM(
                    behavioral_features_of_current_context,means,
                    variances, weights, n, covariance_type)
                if(current_likelihood - previous_likelihood < 0.001):
                    break;

                # Maximization Step
                (means, variances, weights) = do_mstep(
                    behavioral_features_of_current_context,
                    responsibilities, means, variances, weights, n, covariance_type)

                previous_likelihood = current_likelihood

            bayesian_information_criterion = bic(behavioral_features_of_current_context,
                                                 means, variances, weights, n,
                                                 covariance_type)
            if(bayesian_informaiton_criterion < lowest_bic):
                best_means = means
                best_variances = variances
                best_weights = weights
                best_n = n
                best_covariance_type = covariance_type
                lowest_bic =  bayesian_information_criterion
    return (best_means, best_variances, best_weights, best_n, best_covariance_type)
```

Figure 3 Finding Optimal GMM per Context per Subject Pseudo Code

3.3 Testing Phase

During the testing phase, we aim to evaluate how our context-aware behavioral model performs in identifying the correct or the wrong user. We also aim to evaluate how a context-aware behavioral model performs in user identification as compared to a context-agnostic behavioral model.

3.3.1 True Accept

To test the performance of our model, we want to analyze whether our model is able to identify the correct owner (true accept). Ideally, we want the true accept rate to be as high as possible so that the correct user will be prompted to enter secondary authentication a minimum number of times.

To calculate the true accept rate for each subject, we first take all the testing data for that subject. Since each subject has at minimum 1000 data entries and ten percent of those data entries are used for testing, we have at least 100 data entries for testing for each subject. For each data entry in the testing set, we identity the context class of the data entry based on its average cluster distance as shown in the pseudo code below.

```
def get_context_class(contextual_feature):
    min_average_cluster_distance = 1.0 # default to max distance possible
    min_average_cluster_distance_index = 0 #default to the noisy cluster
    for c in range(0, n_clusters+1):
        current_cluster_training_contextual_features =
            total_training_contextual_features[class_member_mask[c]]
        total_cluster_distance = 0
        size_of_current_cluster = current_cluster_training_contextual_features.
            size[0]
        for i in range(size_of_current_cluster):
            total_cluster_distance += get_jaccard_distance(
                current_cluster_training_contextual_features[i], contextual_feature)
        average_cluster_distance = total_cluster_distance / size_of_current_cluster
        if average_cluster_distance < min_average_cluster_distance:
            min_average_cluster_distance = average_cluster_distance
            min_average_cluster_distance_index = c
    return min_average_cluster_distance_index
```

Figure 4 Identifying Context Class of Data Entry Pseudo Code

Once we have found the context class of the test data entry, we use the GMM model established during the training phase for that particular context class to calculate the LPD of the behavioral features extracted from the test data entry. LPD, the log probability density, is simply the log of the probability density of a data entry under a particular context calculated through the GMM model (i.e. LPD = log $(p(x|C))$). In real life, an entry would be accepted as belonging to the authentic user if the LPD is above a certain threshold. We have twelve LPD thresholds (0, -3, -6, -9, -12, -15, -18, -21, -24, -27, -30, -33) to determine whether the test data entry has a higher LPD than the threshold and therefore can be assumed to belong to the actual subject. In our actual study, we use more than twelve thresholds. We select these twelve representative thresholds to show a variety of results in this paper. The reason that we have so many LPD thresholds is to model the different security levels chosen by the user. A user more concerned with security can choose a higher LPD threshold so less similar behavioral patterns in the given context will not be accepted.

After looping through all training data entries of a subject *S*, the true accept rate of the subject *S* with a LPD threshold *t* (0, -3, -6, -9, -12, -15, -18, -21, -24, -27, -30, -33) is defined as the percentage of training data entries identified as the authentic subjects.

3.3.2 False Accept

To test whether our model is robust to adversarial attack, we estimate whether our model might accept "incorrect subjects" (false accept). Optimally, we want the false accept rate to be as low as possible so that the wrong user will not be allowed to continue to use the device.

To find the false accept rate of a subject *S*, we selected ten other subjects randomly. These ten other random subjects served to model the "incorrect subjects". We take the first 100 data entries of each of the other ten "incorrect subjects". The number of data entries used to calculate the false accept rate was chosen to be comparable to the number of data entries used to calculate the true accept rate.

Since all "incorrect subjects" have different sets of Wi-Fi MAC addresses, we expect all data entries of the other ten "incorrect subjects" to belong to the noisy context class C_n of *S*. Indeed, we have to assume this to be true, because the Device Analyzer Dataset has one-way hashed all Wi-Fi MAC addresses and each subject has his unique hash key. As such, we cannot directly compare the sets of Wi-Fi MAC addresses between two subjects. Such a problem will not arise during the actual implementation of our system on an Android app since the same Android device will collect the Wi-Fi MAC addresses for both the authentic user and the adversary who attempts to use the authentic user's phone.

Given the GMM model established during the training phase for C_n, we calculate the LPD of the behavioral features of each of the 100 data entries for each of the other ten subjects. Once

again, we have twelve LPD thresholds (0, -3, -6, -9, -12, -15, -18, -21, -24, -27, -30, -33) to determine whether the test data entry has a higher LPD than the threshold and therefore can be assumed to belong to the subject *S*.

For each of the other ten "incorrect subjects", we define his false accept rate at a LPD threshold *t* (0,-3,-6,-9,-12,-15,-18,-21,-24,-27,-30,-33) to be the percentage of 100 data entries wrongly accepted, meaning, the method incorrectly determined that the "incorrect subject" is *S*. Since we have 10 "incorrect subjects" for each of the 4,600 original subjects, there will be a total of 46,000 false accept rates for the 46,000 "incorrect subjects".

3.3.3 Benchmark: Identification without Context

The hypothesis of our project is that phone usage patterns depend on the location context. Therefore we assume that a context-aware behavioral model would better model the user behaviors and better predict user identity than a context-agnostic behavioral model. To construct a context-agnostic behavioral model, we tweaked the above implementation by assuming all data entries to belong to the same universal context class and building a single GMM for behavioral features across all training data entries of each subject. The true accept rate and false accept rate for this context-agnostic model is similarly calculated based on that single context-independent GMM.

3.4 Model Update

Kayasic et al. mentioned the problem of behavioral drift in the long run [16]. Indeed, a fervent Facebook user might become a zealous Snapchat user after a few months. Not only are usage patterns likely to change in the long run, the location context might shift as well. For example, a student who actively uses his phone at school will likely start using his phone actively at work once he graduates.

We could account for both behavioral drift and contextual drift by updating our context classes and our GMM models based on the new data entries. It should be noted that, in the actual implementation of this system, only authorized data usage would be used to update the context-aware behavioral model continuously. Usage data that first fail to meet the LPD threshold of our context-aware behavioral model and then fail to elicit the correct secondary explicit authentication would simply be discarded. We did not have time, though, to explore the potential value of this approach to model update.

4 Results and Discussions

Based on our context-aware and context-agnostic behavioral models sketched out in the previous section, we evaluated the true accept rate and the false accept rate across all 4,600 subjects. The result of the context-aware model is shown in Section 4.1, while that of the context-agnostic model is shown in section 4.2. Section 4.1 and 4.2 only present a description of the results, while any discussion or interpretation of the description is deferred to Section 4.3.

4.1 Context-aware Behavioral Model

Table 1 on the following page shows the descriptive statistics (mean, minimum, maximum and standard deviation) for the *true accept rate* across all 4,600 subjects using our context-aware behavioral model with different LPD thresholds.

From Table 1, we can see that the mean true accept rate increased significantly as the LPD threshold decreased from 0 to -33. For threshold values between 0 to -9, less than half of the subjects were correctly identified on average. As the threshold decreased below -9, more than half of the subjects were correctly identified on average.

It is also worth noting that the range of the true accept value ranged from 0.0 to 1.0 regardless of LPD threshold value. This shows that our context-aware behavioral model failed to identify certain subjects 100% of the time, while it recognized some other subjects without mistakes. This may indicate the presence of outliers in the dataset, which is explored in more detail in Section 4.3.

Table 1 Descriptive Statistics of True Accept Rate for Context Aware Model

Threshold	Mean	Min	Max	STD
0	0.231	0.0	1.0	0.277
-3	0.313	0.0	1.0	0.294
-6	0.388	0.0	1.0	0.305
-9	0.456	0.0	1.0	0.314
-12	0.514	0.0	1.0	0.318
-15	0.562	0.0	1.0	0.317
-18	0.599	0.0	1.0	0.316
-21	0.628	0.0	1.0	0.313
-24	0.652	0.0	1.0	0.310
-27	0.673	0.0	1.0	0.307
-30	0.691	0.0	1.0	0.303
-33	0.706	0.0	1.0	0.299

Figure 5 below shows the cumulative distributive frequency of the true accept rate based on LPD thresholds from 0 to -33. The X-axis represents the true accept rate ranging from 0.0 to 1.0. The Y-axis represents the number of subjects with a true accept rate lower than or equal to the value indicated on X-axis.

Figure 5 shows that the performance on predicting the correct subject improved as LPD threshold decreased from 0 to -33. A LPD threshold of 0 gave poor performance as around 2,600 subjects were identified as the correct subject less than 20% of the time. A LPD threshold of -33 gave better performance as the number of subjects who were identified correctly less than 20% of the time dropped significantly to 350.

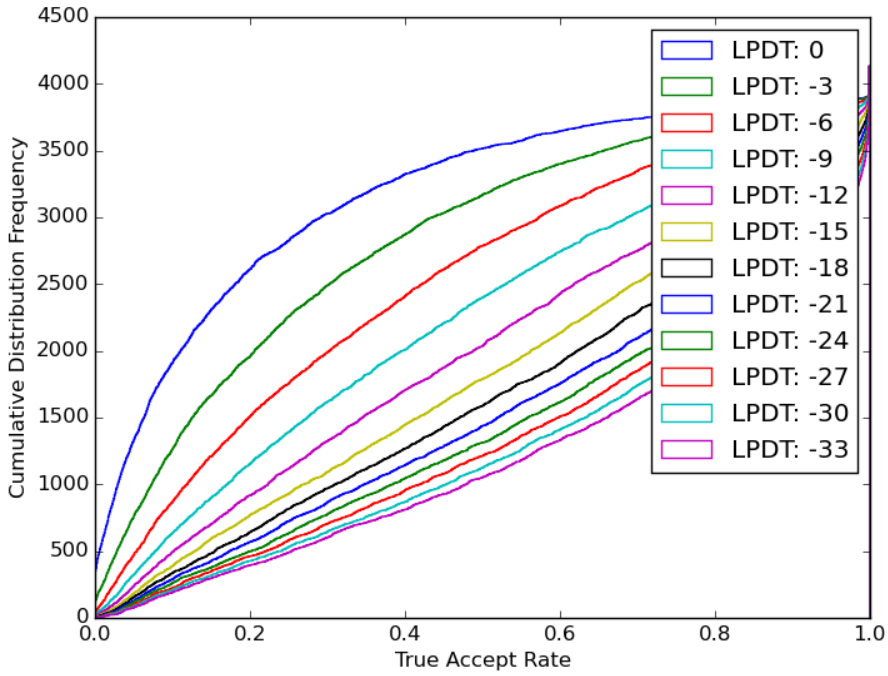

Figure 5 Cumulative Distribution Frequency of True Accept Rate for Context-Aware Model

Table 2 on the next page shows the descriptive statistics (mean, minimum, maximum, and standard deviation) for *the false accept rate* which was calculated for each of the 46,000 "incorrect subjects" using our context-aware behavioral model with different LPD thresholds.

From Table 2, we can see that the mean false accept rate increased slightly as the LPD threshold decreased. However the rate of increase of the mean false accept rate was much slower than the rate of increase of the mean true accept rate as the LPD threshold decreased. Furthermore, the context-aware behavioral model accepted only 6% of the "incorrect subjects" on average even for a LPD threshold as low as -33.

On the other hand, the false accept rate also ranged from 0.0 to 1.0 regardless of the LPD thresholds. This shows that the context-aware behavioral model accepted some "incorrect subjects" 100% of the time while it also rejected some "incorrect subjects" 100% of the time. Again, this indicates a possibility of the presence of outliers in the dataset, which is explored further in Section 4.3.

Table 2 Descriptive Statistics of False Accept Rate For Context Aware Model

Threshold	Mean	Min	Max	STD
0	0.008	0.0	1.0	0.076
-3	0.012	0.0	1.0	0.089
-6	0.016	0.0	1.0	0.104
-9	0.021	0.0	1.0	0.119
-12	0.026	0.0	1.0	0.132
-15	0.031	0.0	1.0	0.146
-18	0.036	0.0	1.0	0.158
-21	0.041	0.0	1.0	0.170
-24	0.046	0.0	1.0	0.181
-27	0.051	0.0	1.0	0.191
-30	0.056	0.0	1.0	0.200
-33	0.060	0.0	1.0	0.209

Similar to Figure 5, Figure 6 shows the cumulative distributive frequency of the false accept rate based on LPD thresholds from 0 to -33. As shown in the figure below, the majority of the "incorrect subjects" were accepted close to 0% of the time for LPD thresholds from 0 to -33. While Figure 5 shows a huge improvement in performance on identifying the correct subject as the LPD decreased, Figure 6 shows only a slight drop in performance on filtering out the incorrect subjects as the LPD decreased.

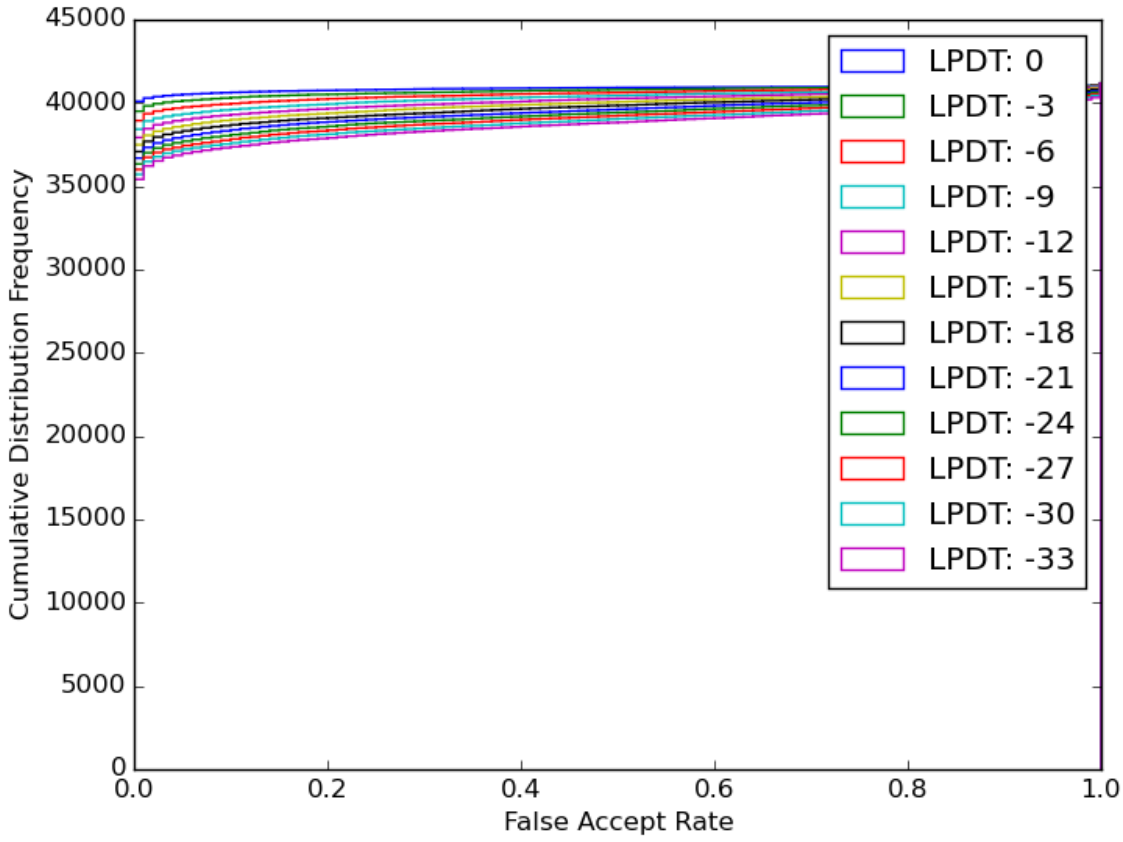

Figure 6 Cumulative Frequency of False Accept Rate for Context-Aware Model

4.2 Context-agnostic Behavioral Model

Table 3 on the following page shows the descriptive statistics (mean, minimum, maximum, and standard deviation) for *the true accept rate* across all 4,600 subjects using our context-agnostic behavioral model with different LPD thresholds.

As in the context-aware model, the context-agnostic model generated higher mean true accept rate as the LPD threshold decreased from 0 to -33. Furthermore, the context-agnostic model also generated true accept rate which ranged from 0.0 to 1.0 regardless of the LPD threshold. This shows that our context-agnostic behavioral model also failed to identify certain subjects 100% of the time, while it recognized some other subjects without mistakes.

Compared to the context-aware model, the context-agnostic model generated higher mean true accept rate at each value of LPD threshold. In fact, at LPD threshold of -33, the context-agnostic model identified the correct user close to 90% of the time on average, while the context-aware model identified the correct user only 70% of the time on average. The observation that the context-agnostic model identified the correct subjects more frequently than the context-aware model did is also shown in Figure 7. Figure 7 on the page after shows the cumulative distribution frequency of the true accept rate for both the context-aware and the context-agnostic models at the LPD threshold of -33. As shown in Figure 7, the number of subjects being correctly identified less than 20% of the time by the context-aware model was more than double that by the context-agnostic model.

Table 3 Descriptive Statistics of True Accept Rate for Context Agnostic Model

Threshold	Mean	Min	Max	STD
0	0.334	0.0	1.0	0.332
-3	0.468	0.0	1.0	0.337
-6	0.573	0.0	1.0	0.332
-9	0.660	0.0	1.0	0.320
-12	0.728	0.0	1.0	0.303
-15	0.777	0.0	1.0	0.305
-18	0.811	0.0	1.0	0.273
-21	0.836	0.0	1.0	0.260
-24	0.856	0.0	1.0	0.248
-27	0.871	0.0	1.0	0.237
-30	0.883	0.0	1.0	0.228
-33	0.893	0.0	1.0	0.220

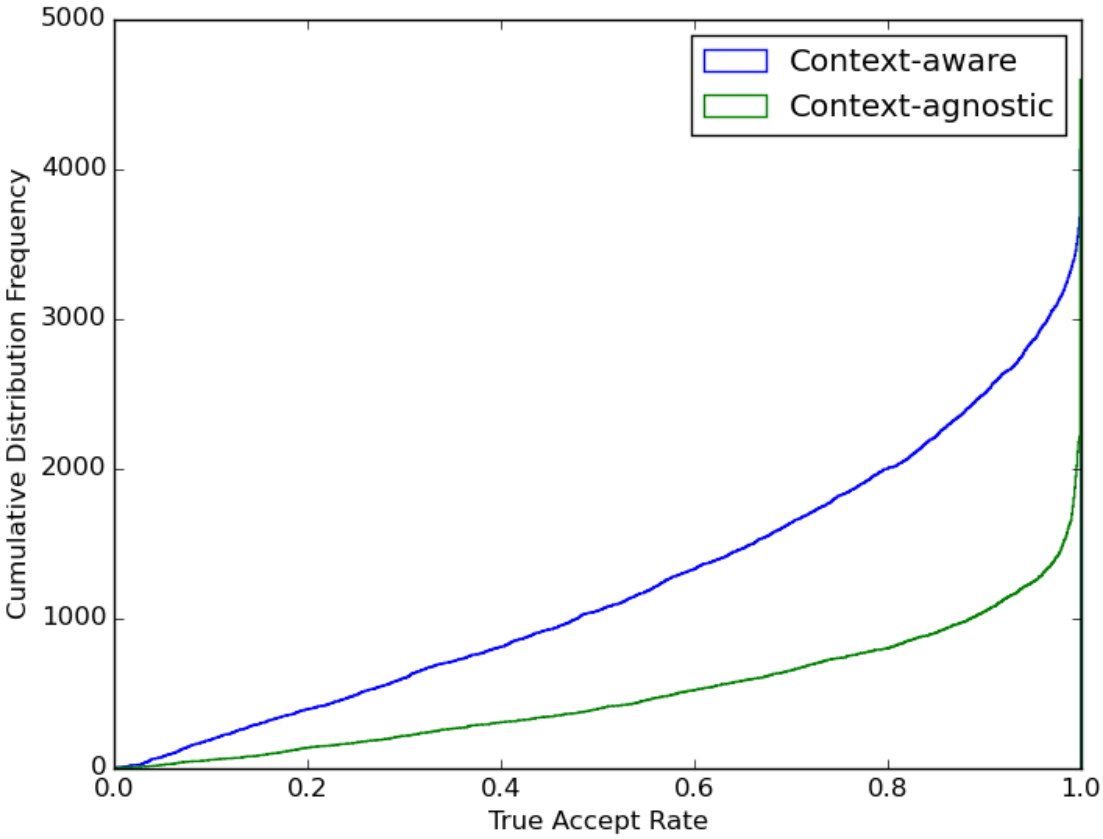

Figure 7 Cumulative Distribution Frequency of True Accept Rate at LPDT -33

Table 4 on the next page shows the descriptive statistics (mean, minimum, maximum, and standard deviation) for *the false accept rate* across all subjects using our context-agnostic behavioral model with different LPD thresholds.

As in the context-aware model, context-agnostic model had higher mean false accept rate as the LPD threshold decreased. The rate of increase of the mean false accept rate was also slower than the rate of increase of the mean true accept rate as the LPD threshold decreased. Further, all LPD threshold generated the false accept rate which ranged from 0.0 to 1.0. This shows that the context-agnostic behavioral model also accepted some "incorrect subjects" 100% of the time while it also rejected some "incorrect subjects" 100% of the time.

While the context-agnostic model had a higher mean true accept rate at each LPD threshold value than the context-aware model, the context-agnostic model also generated smaller false accept rate at each LPD threshold value as well. For example, at the LPD threshold of -33, the context-agnostic model accepted 13.2% the "incorrect subjects" on average while the context-aware model accepted only 6%. The observation that the context-agnostic model filtered out the "incorrect subjects" less frequently than the context-aware model did is also shown in Figure 8 on the page after. Figure 8 shows the cumulative distribution frequency of the false accept rate for both the context-aware and the context-agnostic models at the LPD threshold of -33. As shown in Figure8, the number of "incorrect subjects" being accepted close to 100% of the time by the context-agnostic model was more than that by the context-aware model.

In fact, the context-agnostic model and the context-aware model generated similar pairs of true accept rate and false accept rate at different LPD threshold value. For example, context-agnostic model generated a true accept rate of 0.728 and a false accept rate of 0.055 at a LPD threshold of

-12, while the context-aware model generated a true accept rate of 0.691 and a false accept rate of 0.055 at a LPD threshold of -30. Such observation shows that the addition of context information did not contribute to a better behavioral model. This observation is explored further in Section 4.3

Table 4 Descriptive Statistics of False Accept Rate for Context Agnostic Model

Threshold	Mean	Min	Max	STD
0	0.014	0.0	1.0	0.100
-3	0.021	0.0	1.0	0.121
-6	0.030	0.0	1.0	0.145
-9	0.042	0.0	1.0	0.172
-12	0.055	0.0	1.0	0.199
-15	0.069	0.0	1.0	0.223
-18	0.081	0.0	1.0	0.243
-21	0.093	0.0	1.0	0.260
-24	0.104	0.0	1.0	0.274
-27	0.114	0.0	1.0	0.287
-30	0.123	0.0	1.0	0.298
-33	0.132	0.0	1.0	0.308

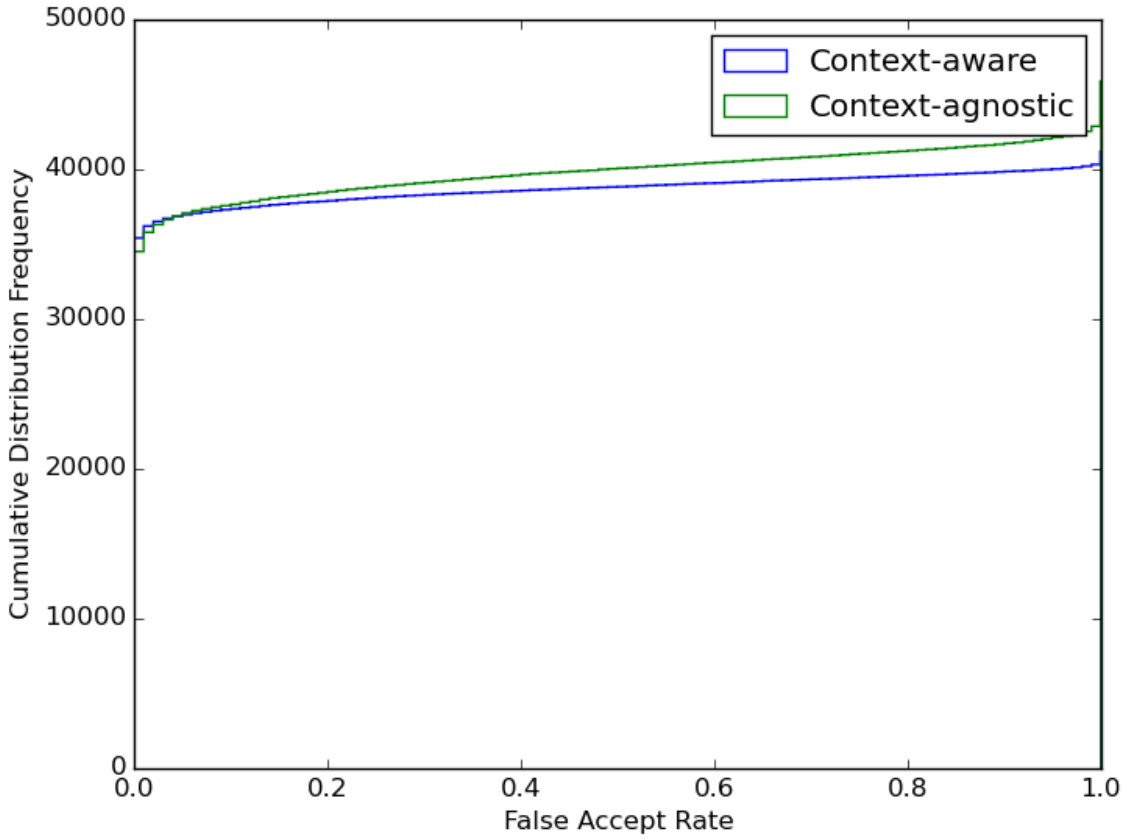

Figure 8 Cumulative Distribution Frequency of False Accept Rate at LPDT -33

4.3 Discussion

With regards to our aim to develop a continuous authentication, both our context-aware and context-agnostic models are viable schemes. For the context-aware model, a LPD threshold of -33 authorized the correct user more than 70% of the time while it authorized the incorrect user only 6% of the time. During the 30% of the time when our context-aware model rejected the correct user, the phone could possibly prompt the user with an initial authentication method so that the correct user could continue to use his phone and the wrongly labeled data entry could then be used to update and improve our context-aware behavioral model.

While a 70% true accept rate might seem to entail a lot of re-logins, such result is actually comparable to results reported by relevant research [9]. Furthermore, decreasing the LPD threshold even further below -33 can lead to a higher true accept rate and less trouble of re-logins. For example, a LPD threshold of -66 generated 80.6% true accept rate. However, the false accept rate also rose to 10.0% at a LPD threshold of -66. Ultimately, the setting of LPD threshold would determine the fine balance between convenience (fewer re-logins for authentic users) and security (fewer incorrect users accepted). How to set the appropriate LPD threshold would be explored in future work.

With regards to our aim of developing a hierarchical authentication method, both our context-aware and context-agnostic behavioral models demonstrate the possibility of satisfying a hierarchy of users' security needs. Since both the false accept rate and the true accept rate increased with decreasing LPD threshold, it is possible to allow the user to specify different LPD thresholds to correspond to different security needs. A banking app with sensitive content might need a higher (i.e., less negative) threshold than a weather app with no personal content. The

higher threshold for the banking app will block most of the incorrect users (low false accept rate), but it will also require more frequent re-logins for the correct users (low true accept rate). The reverse is true for the low threshold for the weather app. Once again, the determination of the appropriate LPD thresholds for different apps requires further exploration.

Contrary to our assumption, the context-agnostic and the context-aware behavioral models have similar performances in identifying the correct or the incorrect user, albeit at different LPD threshold value. One reason may be a suboptimal selection for the contextual features and the context clustering algorithm. Either the location context itself or the location context clusters found by DBSCAN might not be indicative of different app-usage patterns. Figure 9 below shows a histogram of the number of clusters found by DBSCAN for each of the 4,600 subjects. It is clear from Figure 9 that a vast majority of subjects have only one or two context classes identified by DBSCAN. The small number of context clusters identified might partially explain why context-agnostic and context-aware models showed similar performances.

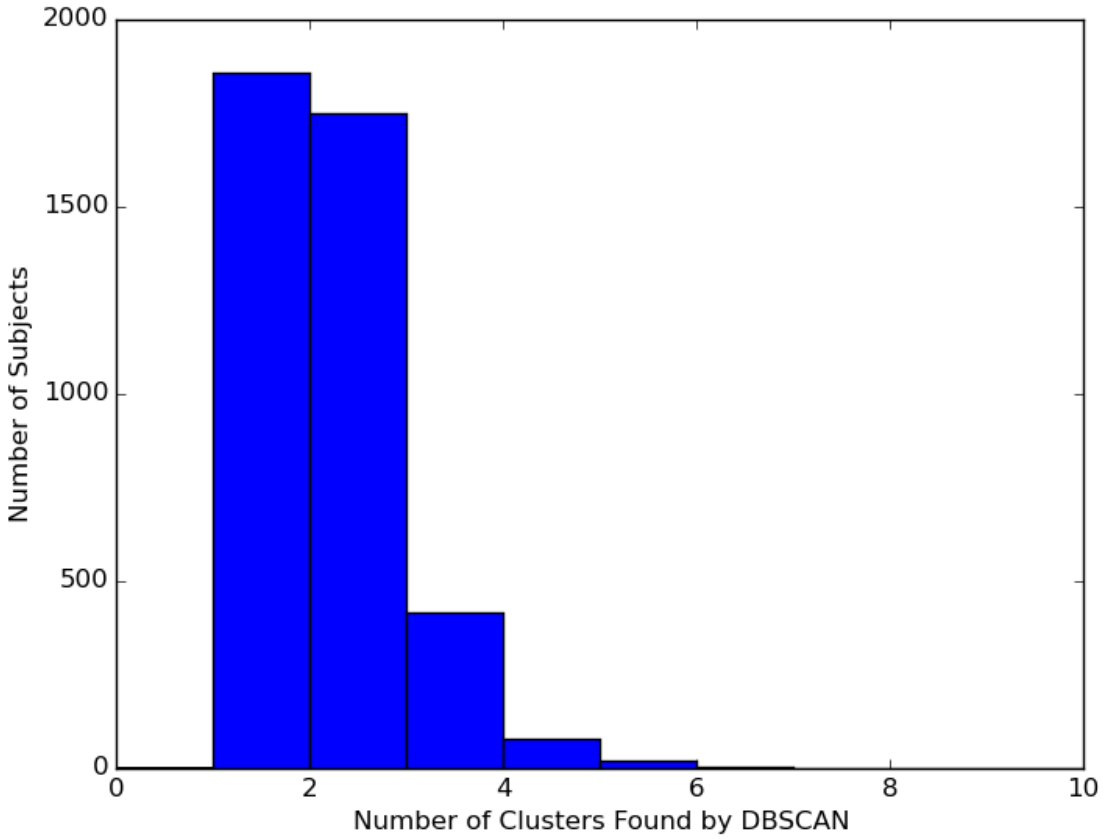

Figure 9 Histogram of Number of Clusters Found by DBSCAN

There may be other reasons for the unsatisfactory performance of the context-aware model. First, the context-dependent behaviors might not be properly modeled by GMM. A different behavioral model might generate better results for the context-aware model than the context-agnostic model. Second, the data collection of the Device Analyzer Dataset is entirely voluntary and therefore sporadic. Some subjects have huge gaps between two consecutive data collections. Therefore a Wi-Fi scan may be very far away from its closest subsequent incidence of app-usage data collection. The misalignment between the contextual data and the behavioral data might explain why context did not contribute to a more accurate behavioral model. We could possibly

eliminate this problem by having a stricter selection of qualified subjects, who have more consistent data collection.

In our experiments, we found some subjects with a true accept rate as low as 0.0, and others with a false accept rate as high as 1.0. That is, there were certain outliers among the subjects who could not be properly modeled by our context-aware or context-agnostic behavioral model. Such outliers may arise because we divided the 21-day of data sequentially into training and testing set. It is possible that the testing set happened to correspond to a period when the subject behaved abnormally. Those outliers may also arise due to the lack of data for that subject. In fact, our minimum requirement of 1,000 data entries, if collected at 5-minutes intervals, corresponds to just 3.5 days of data out of the 21-day period of data collection. Figure 10 below shows a 6-degree polynomial fit of a scatter plot of the true accept rate against the number of data entries of each of the 4,600 subjects. The true accept rate was calculated based on the context-aware model using a LPD threshold of -33. It is clear that an increase in the number of data entries generally led to an increase in the true accept rate. But, there is a dip in the true accept rate around 7,000 data entries. As shown in Appendix I, such a dip may be due to a lack of subjects who have more than 7,000 data entries.

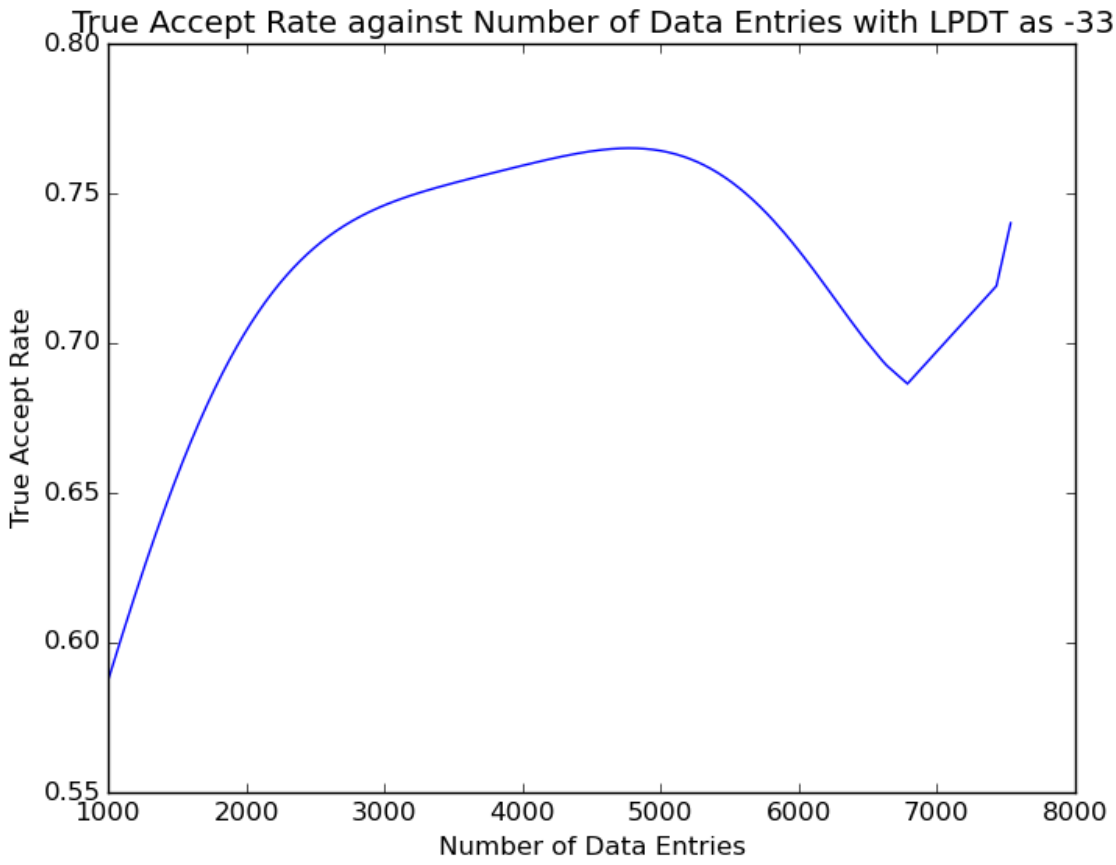

Figure 10 Number of Data Entries against True Accept Rate for Context-Aware Model at LPDT -33

5 Future Work

In the future, we need to investigate why location context did not contribute to a better behavioral model as we had originally proposed. We also need to find out a systematic way of determining the LPD thresholds for different apps and under different contexts. In the end, we propose a hypothetical Android authentication app based on our scheme.

5.1 Further Investigation of Our Assumption

As discussed in Section 4.3, the context-aware and the context-agnostic models yielded similar results, which is contrary to our initial assumption that user behaviors depend on contexts and context-aware model should be more accurate. To investigate why our initial assumption did not hold, we would experiment with other behavioral and contextual features, different context classification algorithms, and more nuanced behavioral models. More experiments would hopefully tell us whether our initial assumption is indeed wrong or our choice of algorithm was suboptimal and misled our conclusion.

5.1.1 Other Behavioral and Contextual Features

In our approach, we assume the app-usage patterns to be dependent on the location context indicated by Wi-Fi MAC addresses. If we have more data on GPS in the future, we could experiment with GPS coordinates as an indicator of location context. We could also experiment with different features to augment Wi-Fi MAC addresses to find better context classes that are more indicative of different app-usage patterns. For example, the physical activity level of the smartphone users might be a good indicator of usage context since users are likely to use

different apps when they are still as compared to when they are moving. The activity level can easily be extracted from sensor data.

Our behavioral model has relied exclusively on app-usage data such as the duration of foreground app-usage, the number of apps used, the duration of usage of the top three apps, and the time of the day. A more accurate model of app-usage pattern might include more features such as the ratings of the apps at foreground as reported by Google Play, the date of last update of the apps at foreground, and the category of apps used at foreground as given by Google Play. We could also experiment with the duration of usage of the top five, top seven, or maybe top ten apps.

Besides experimenting with other behavioral and contextual features, we also need to ensure each incidence of behavioral data collection aligns with its corresponding incidence of contextual data collection. As mentioned in Section 4.3, a Wi-Fi scan might be far away from its subsequent app-usage data collection. The misalignment between the contextual data and the behavioral data could be eliminated by having a stricter selection of qualified subjects who should not have large gaps between consecutive data collections on context and behaviors.

Furthermore, we could experiment with setting a higher requirement for the minimum number of data entries for the qualified subjects. As discussed in Section 4.3, the lack of data entry for some subjects might lead to the presence of outliers. As such, selecting subjects with more than, say 2,000, data entries might help alleviate the problem of outliers and build a better context-aware behavioral model

5.1.2 Better Context Classification

As discussed in Section 4.3, the current context clustering algorithms found only 1 or 2 contexts for the majority of the subjects. We could experiment with a better clustering algorithm for contextual features. For example, a hierarchical clustering algorithm could be used instead of DBSCAN. It is also possible to keep the DBSCAN algorithm and experiment with different distance functions for Wi-Fi MAC addresses and different *epsilon* and *min_samples* value.

5.1.3 Better Behavioral Model

Currently, we use one GMM per context to model the usage patterns. We would explore a Hidden Markov Model (HMM) to model the context-dependent behavioral patterns. In our proposed HMM, different context classes would represent different hidden states. Each context class would have different transition probabilities to other context classes and to itself. Each context class would have emission probabilities modeled by GMM for different behavioral features. Both transition probabilities and perimeters of GMM for emission probabilities would be found through expectation-maximization.

5.2 Determination of LPD Threshold

As discussed in Section 4.3, a higher LPD threshold means more security (fewer "incorrect subjects" accepted) but less convenience (more frequent re-logins for authentic users), while a lower LPD threshold means less security but more convenience. Ideally, we should have a *high LPD threshold* for apps with sensitive contents (e.g. Banking, HealthCare, and Shopping) and for contexts that are vulnerable to adversarial attacks (e.g. unfamiliar contexts). In contrast, we

should have a *low LPD threshold* for apps without personal information (e.g. Weather, Alarm, and News) and for contexts where thefts are less likely (e.g. home).

To find out the exact value of the high and low LPD thresholds, we could conduct a survey among 100 people to ask them what the acceptable ranges of the true accept rate and the false accept rate are for different apps and under different contexts. We would then set the same LPD thresholds for all users according to the survey results. For example, if the majority of the survey participants indicated that they want their Banking app to be never available to strangers (i.e. close to 0% false accept rate) and they would be fine having multiple re-logins for this Banking app (i.e. low true accept rate), we would set the LPD threshold for this Banking app for all phone users at 0, which has 0.8% false accept rate and 23.1% true accept rate.

Besides setting a generic LPD threshold for all users, we could also experiment with setting different LPD thresholds for different users. For example, we could survey each phone user who chooses to use our scheme. We would ask each user about the acceptable ranges of the true accept rate and the false accept rate for each app on the phone and for all common contexts. If one phone user indicates that he prefers to have fewer re-logins even for the Banking app, we could set the LPD threshold for the Banking app on his phone to be -33, which has 70% true accept rate and 6% false accept rate.

5.3 The Android Authentication App

We could use our context-aware behavioral model to build an Android app that would take into considerations the hierarchy of security needs of smartphone users and continuously authenticate users as they are using the phones. The overall structure and some considerations of this hypothetical app are described as follows.

5.3.1 App Overview

The authentication app would passively collect contextual and behavioral features in the background. For the first three weeks after installation, the app would train a context-aware behavioral model for the user. After the first three weeks, the app would use the developed model to authenticate the user continuously. The authentication could be either system-wide or app-specific. Based on the continuously calculated LPD of the user's behaviors in contexts, the user would be locked out of the phone usage if his LPD is lower than the threshold specified for the phone or he would not gain access to certain sensitive apps which have a higher LPD threshold than his calculated LPD. All authorized usage after the first three weeks would be used to continuously update the model.

5.3.2 Sampling Frequency

One consideration for the deployment of the app is how frequently the app should collect the contextual and behavioral features from the phone. A sampling frequency that is too high would drain the battery fast. A sampling frequency that is too low would not adequately capture behavioral changes between two data collection incidences nor identify the correct and incorrect users timely.

The Device Analyzer Dataset has an unknown sampling frequency. A close inspection of some data files reveals a sampling frequency of about one sample per five minutes. Such sampling rate is too low in our opinion, as it would take five minutes for the model to identify the incorrect user and much sensitive information would have already been stolen during this five-minute interval. Ideally, we want out sampling frequency to be one sample per few seconds so the incorrect user would be identified and locked out immediately.

5.3.3 Automatic Switching from Training to Deployment

Our proposed approach for the authentication app specifies that the app would train a context-aware behavioral model and start authenticating users three weeks after installation. The choice of three weeks is entirely heuristic, however. Kayacik et al. have suggested that the app should be able to automatically switch from training to deployment once the behavioral model is sufficiently stable [16]. We could alternatively build our model incrementally based on each incoming data entry. Once the model has minimum change between two consecutive data updates, the app should start authenticating the users.

5.3.4 Storage and Computation: Local or the Cloud

Another concern for this authentication app is where the context-aware behavioral model should be computed and stored. Training and storing the behavioral model in the cloud might provide faster training and authentication time given the resources available in the cloud. The cloud-based approach would also free up space on the smartphones [9]. However the cloud-based approach might incur too much latency if the phone must access the remote model each time it needs to authenticate a user. Furthermore, storing sensitive behavioral and contextual information on the cloud might also raise privacy risks.

5.3.5 Model Update

In the future, we need to study ways to incrementally update our context classes and behavioral models based on incoming data during periods of high authentication confidence.

5.3.6 Adversarial Model

As future work, we could also explore a different adversarial model for our authentication app. Kayacik et al. suggested four attack case studies: “uninformed outsider”, “informed outsider”, “uninformed insider”, and “informed insider” [16]. In our case, an “uninformed outsider” would be an incorrect user with different contexts and different behaviors. An “informed outsider” would be an incorrect user with different contexts but similar behaviors. An “uninformed insider” would be an incorrect user with similar contexts but different behaviors. An “informed insider” would be an incorrect user with similar contexts and similar behaviors.

To calculate the false accept rate, we evaluated the LPD of the first 100 data entries of 10 other randomly selected subjects for each original subject. We assumed all 100 data entries belonged to the noisy context of the original subject (different contexts), and we picked the other 10 subjects randomly so the behaviors of the 10 subjects might be different from those of the original subject (different behaviors). As such, our adversarial model is essentially an “uninformed outsider” model. In the future, we would investigate other adversarial models.

6 Summary and Conclusion

We proposed a new scheme for continuous and hierarchical authentication on the smartphones. The key assumption of our scheme was that the app-usage patterns are dependent on location context. We built both a context-aware behavioral model and a context-agnostic behavioral model. We tested both models on 4,600 subjects from the Device Analyzer Dataset. Our experimental results indicated that both the context-aware and the context-agnostic behavioral models are viable continuous and hierarchical authentication schemes. Contrary to our initial assumption, the addition of location context does not contribute to an improvement in the performance of our behavioral model. Future work could explore why our assumption did not hold and how to set the LPD thresholds systematically. An Android authentication app would also be possible based on our model.

7 Acknowledgements

First and foremost, I would love to thank my thesis advisor, David Kotz. He has been a tremendously helpful and inspiring mentor throughout my first adventure of writing a research paper. I could have written a very generic paragraph about how Professor Kotz has taught me the important techniques of sound research and effective communication. However, I prefer to take this opportunity to write down three most memorable moments during my senior thesis research with Professor Kotz. The first moment was when Professor Kotz finally approved my research poster which I was going to present at Johns Hopkins University. This final draft was in fact the SEVENTH draft of my poster, which I spent weeks editing and improving on. For every single draft, Professor Kotz provided valuable and detailed suggestions. Those suggestions not only helped me make huge improvements on my poster, but also taught me how to communicate my ideas in an effective and succinct way. The second moment was when I just finished presenting my poster at Johns Hopkins University and Professor Kotz came quietly to my seat to give me a thumb-up. The encouragement made me feel so proud of my own work and energized me to do even better at my senior thesis. The last moment was when Professor Kotz talked about the possibility of publishing this paper if more work could be accomplished. Never in my life would I have imagined the publication of my OWN paper. I believe this is preciously the trademark of a great mentor, who broadens people' horizons and help them see the many possibilities in life.

Next, I would love to thank Dr. Andrés Molina-Markham for the initial idea of this project. Dr. Molina-Markham helped me to develop the initial codes on this project. He also gave me expert technical advice throughout my senior thesis research.

Many thanks also go to Dr. Alina Oprea for her technical suggestions and unwavering moral support throughout the research.

I would also love to thank Mr. Shrirang Mare for helping me with the initial downloads of the Device Analyzer Dataset. I would love to thank Mr. Tim Tregubov and Mr. Yoonjoo Choi for helping me to use the Anthill cluster.

I sincerely appreciate the Device Analyzer Dataset provided by Cambridge University.

I sincerely thank RSA Laboratories for supporting this research project in its initial stages.

This research results from a research program at the Institute for Security, Technology, and Society at Dartmouth College, supported by the National Science Foundation under award number CNS-1329686. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

Appendix I Data Quality of the 4,600 Subjects

We selected only those 4,600 subjects in the Device Analyzer Dataset who have at least 1,000 data entries for their first 21 days of data. Figure 11 below shows the histogram of number of data entries (must be larger than 1000) for each of the 4,600 subjects. Figure 12 below shows the histogram of number of days of data collection (must be smaller than 22) for each of the 4,600 subjects. Figure 13 below shows the histogram of the number of data entries per day for each of the 4,600 subjects.

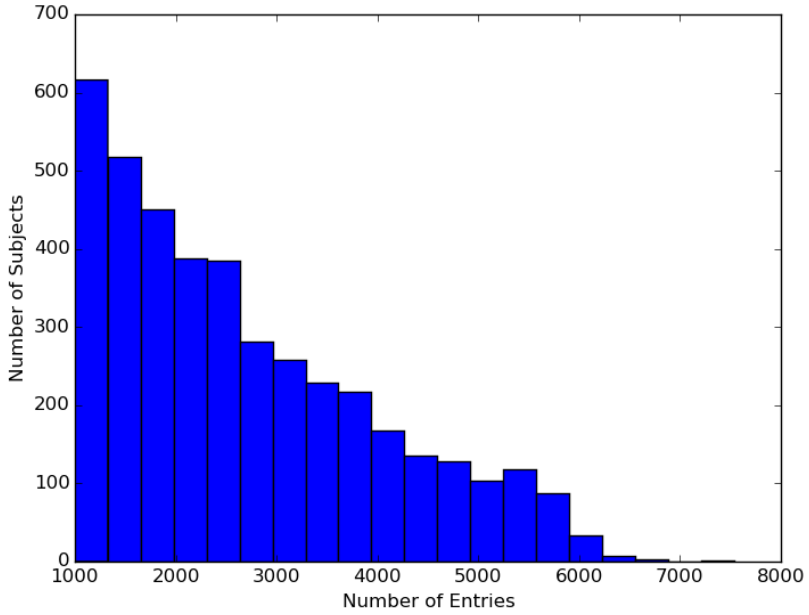

Figure 11 Histogram of Number of Entries per Subject

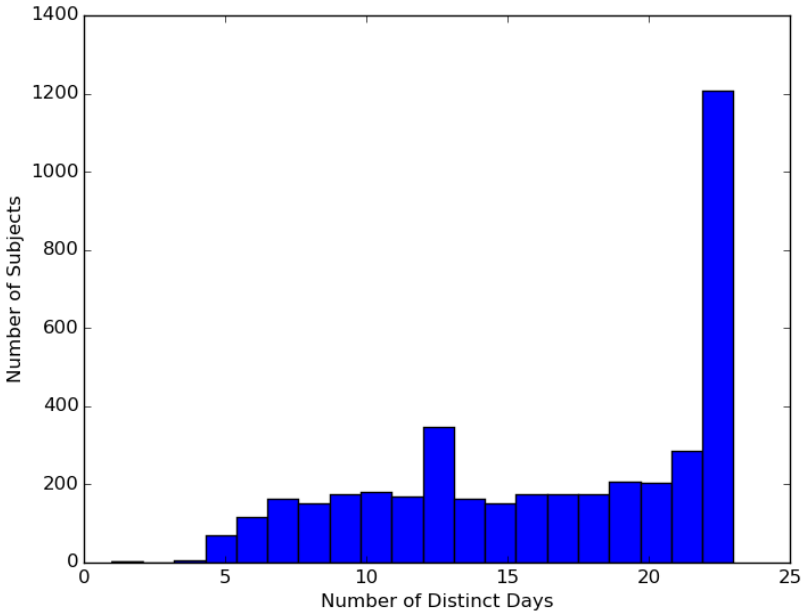

Figure 12 Histogram of Number of Days of Data Collection per Subject

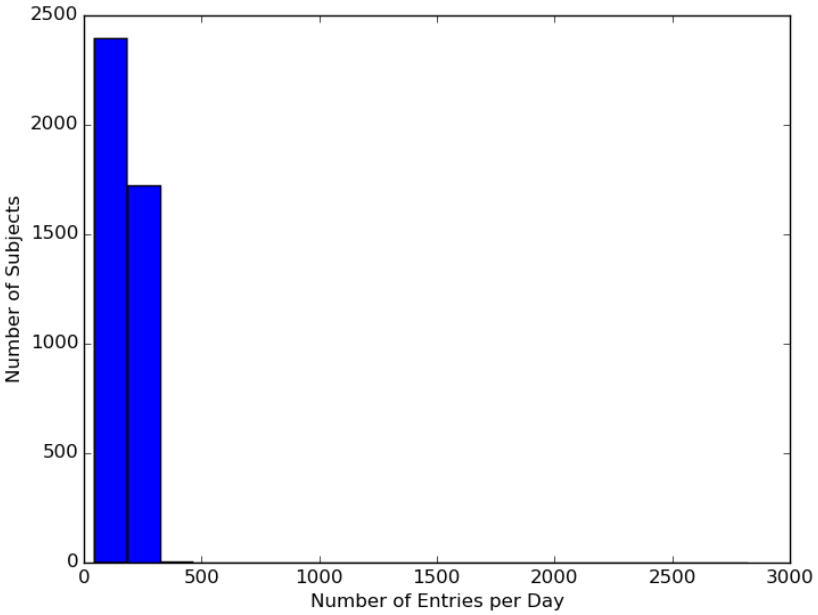

Figure 13 Histogram of Number of Data Entries per Day per Subject

Appendix II DBSCAN Clustering Results

To select appropriate values for *epsilon* and *min_samples*, we selected 20 representative subjects who had at least 1,000 entries of Wi-Fi MAC addresses and GPS coordinates for the first 21 days of data collection. For each subject, we clustered his Wi-Fi MAC addresses using *epsilon* values increasing from 0.1 to 0.9 at an interval of 0.2, and *min_samples* values increasing from 60 to 150 at an interval of 30. We define the best *epsilon* and *min_samples* value pair to be the one which generated relatively fewer "noisy points" and produced a reasonable number of clusters (e.g. 1~5).

Figure 14 shows the descriptive statistics (mean, minimum, and maximum) for the number of clusters for the 20 representative subjects against the five different values of *epsilon* (0.1, 0.3, 0.5, 0.7, 0.9). Figure 15 shows the descriptive statistics (mean, minimum, and maximum) for the number of "noisy points" for the 20 representative subjects against the five different values of *epsilon* (0.1, 0.3, 0.5, 0.7, 0.9).

Figure 16 shows the descriptive statistics (mean, minimum, and maximum) for the number of clusters for the 20 representative subjects against the four different values of *min_samples* (60, 90, 120, 150). Figure 17 shows the descriptive statistics (mean, minimum, and maximum) for the number of "noisy points" for the 20 representative subjects against the four different values of *min_samples* (60, 90, 120, 150).

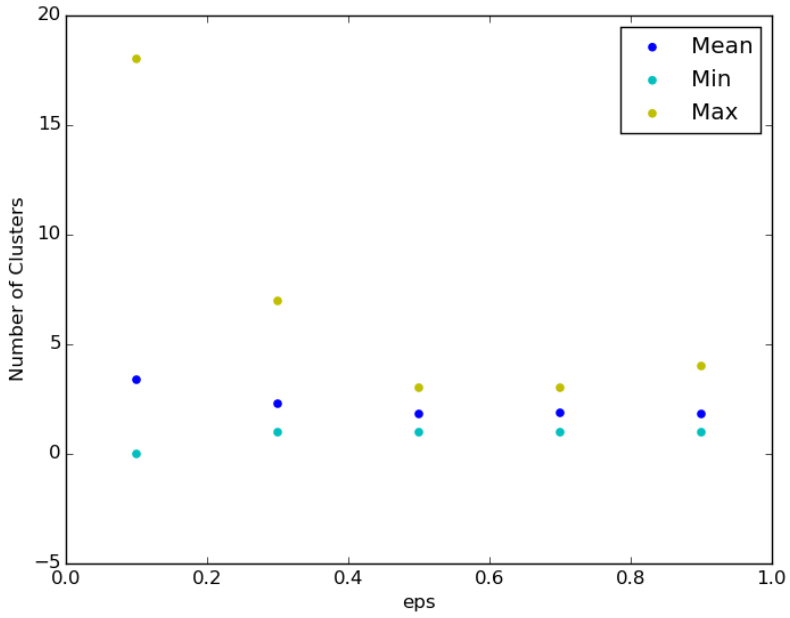

Figure 14 Number of Clusters Against EPS for 20 Representative Subjects

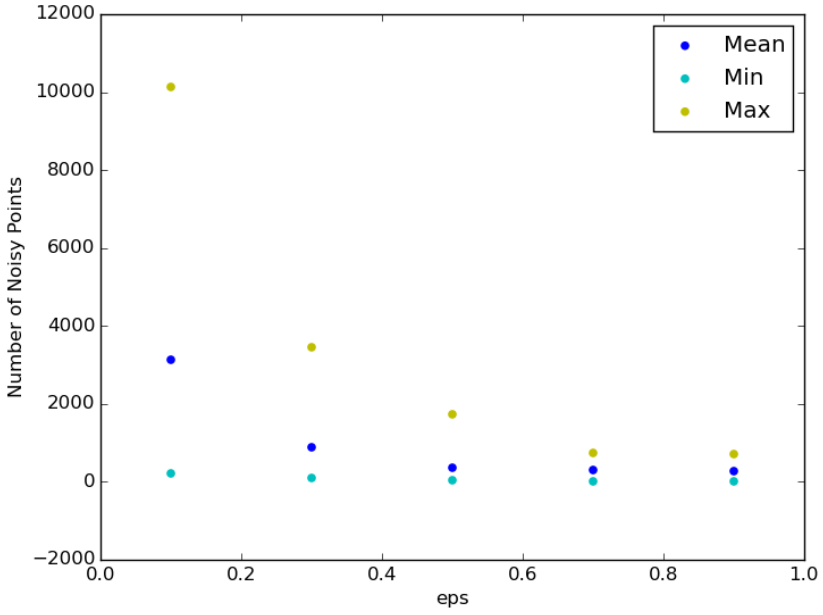

Figure 15 Number of Noisy Points Against EPS for 20 Representative Subjects

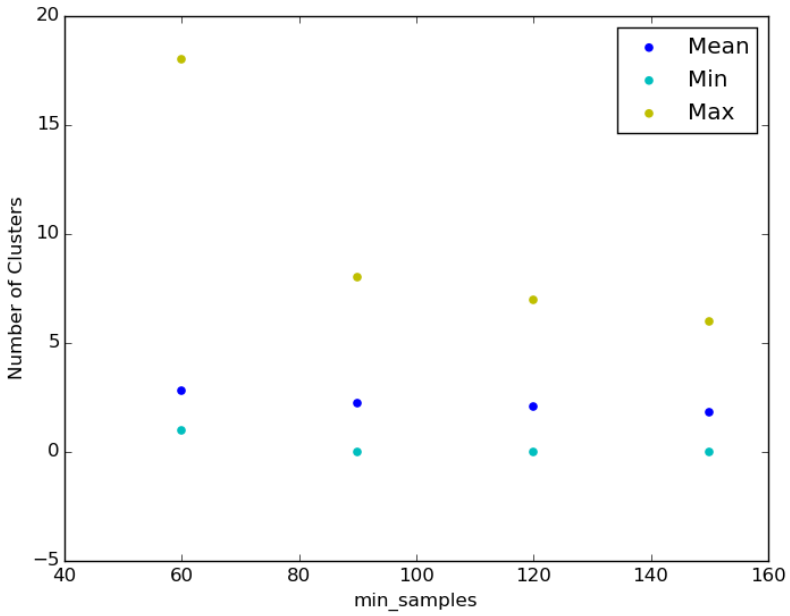

Figure 16 Number of Clusters Against MIN_SAMPLES for 20 Representative Subjects

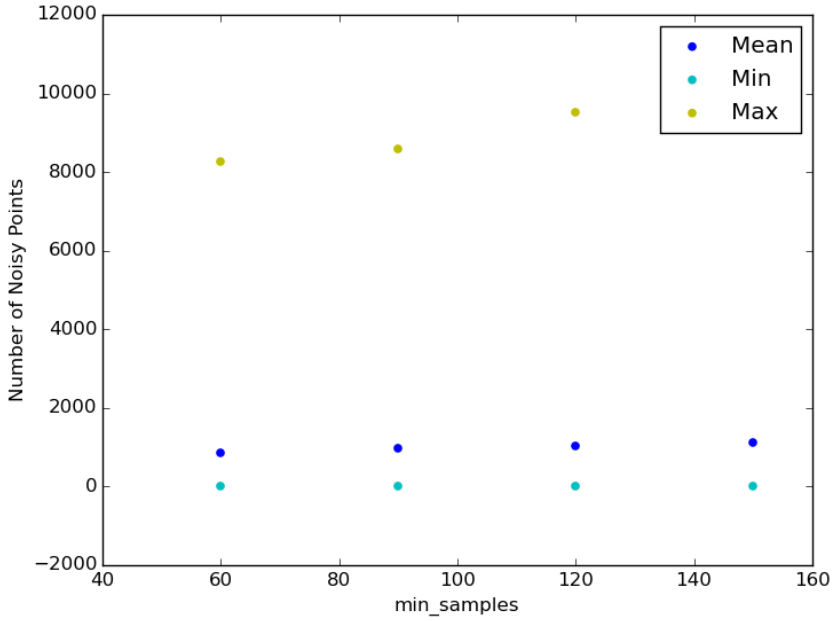

Figure 17 Number of Noisy Points Against MIN_SAMPLES for 20 Representative Subjects

Bibliography

[1] eMarketer, and AP. *Number of smartphone users worldwide from 2014 to 2019 (in millions).* http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (accessed February 22, 2016).

[2] Hayashi, Eiji, Oriana Riva, Karin Strauss, A. J. Brush, and Stuart Schechter. "Goldilocks and the two mobile devices: going beyond all-or-nothing access to a device's applications." In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, no. 2. ACM, 2012.

[3] Clarke, Nathan L., and Steven M. Furnell. "Authentication of users on mobile telephones–A survey of attitudes and practices." In *Computers & Security*, vol 24, no. 7 pp. 519-527. Elsevier, 2005.

[4] Holz, Christian, and Marius Knaust. "Biometric Touch Sensing: Seamlessly Augmenting Each Touch with Continuous Authentication." In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 303-312. ACM, 2015.

[5] Shi, Elaine, Yuan Niu, Markus Jakobsson, and Richard Chow. "Implicit authentication through learning user behavior." In *Information security*, pp. 99-113. Springer Berlin Heidelberg, 2010.

[6] Frank, Jordan, Shie Mannor, and Doina Precup. "Activity and Gait Recognition with Time-Delay Embeddings." In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence,* no. 10, pp 1581-1586. 2010.

[8] Clarke, Nathan L., and S. M. Furnell. "Authenticating mobile phone users using keystroke analysis." In *International Journal of Information Security* 6, no. 1 pp. 1-14. 2007.

[8] Hayashi, E., Das, S., Amini, S., Hong, J. and Oakley, I. “CASA: context-aware scalable authentication.” In *Proceedings of the Ninth Symposium on Usable Privacy and Security*. No 3. ACM. 2013.

[9] Riva, Oriana, Chuan Qin, Karin Strauss, and Dimitrios Lymberopoulos. "Progressive authentication: deciding when to authenticate on mobile phones." *In USENIX Security Symposium*, pp. 301-316. 2012.

[10] Khan, Hassan, Aaron Atwater, and Urs Hengartner. "A comparative evaluation of implicit authentication schemes." In *Research in Attacks, Intrusions and Defenses*, pp. 255-275. Springer International Publishing, 2014.

[11] Feng, Tao, Jun Yang, Zhixian Yan, Emmanuel Munguia Tapia, and Weidong Shi. "TIPS: context-aware implicit user identification using touch screen in uncontrolled environments." In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, p. 9. ACM, 2014.

[12] Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A density-based algorithm for discovering clusters in large spatial databases with noise." In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, vol. 96, no. 34, pp. 226-231. 1996.

[13] Jaccard, Paul. "The distribution of the flora in the alpine zone." In *New Phytologist* 11, no. 2 pp.37-50. 1912.

[14] Scikit Learn, *Cluster.DBSCAN*, http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (accessed on 2/27/2016)

[15] Scikit Learn, *Cluster.GMM* https://github.com/scikit-learn/scikit-learn/blob/51a765a/sklearn/mixture/gmm.py#L579 (accessed on 2/27/2016)

[16] Kayacik, Hilmi Gunes, Mike Just, Lynne Baillie, David Aspinall, and Nicholas Micallef. "Data driven authentication: On the effectiveness of user behaviour modelling with mobile device sensors." arXiv:1410.7743. 2014.

[17] Cambridge University, *Device Analyzer Dataset*, https://deviceanalyzer.cl.cam.ac.uk/ (accessed on 2/27/2016)